



Objectives

- Learn more Trigger concepts
- Create additional database triggers
- Explain the rules governing triggers
- Implement triggers



Creating Triggers on System Events

```
CREATE [OR REPLACE] TRIGGER trigger_name timing
[database_event1
[OR database_event2 OR ...]]
ON {DATABASE|SCHEMA}
trigger_body
```



Log On and Log Off Trigger Example

```
SQL> CREATE OR REPLACE TRIGGER LOGOFF_TRIG

2 BEFORE logoff ON SCHEMA

3 BEGIN

4 INSERT INTO log_trig_table

5 (user_id, log_date, action)

6 VALUES (user, sysdate, 'Logging off');

7 END;
```



CALL Statement

```
CREATE [OR REPLACE] TRIGGER trigger_name

timing

event1 [OR event2 OR event3]

ON table_name

[REFERENCING OLD AS old | NEW AS new]

[FOR EACH ROW]

[WHEN condition]

CALL procedure_name
```

```
SQL> CREATE TRIGGER TEST3

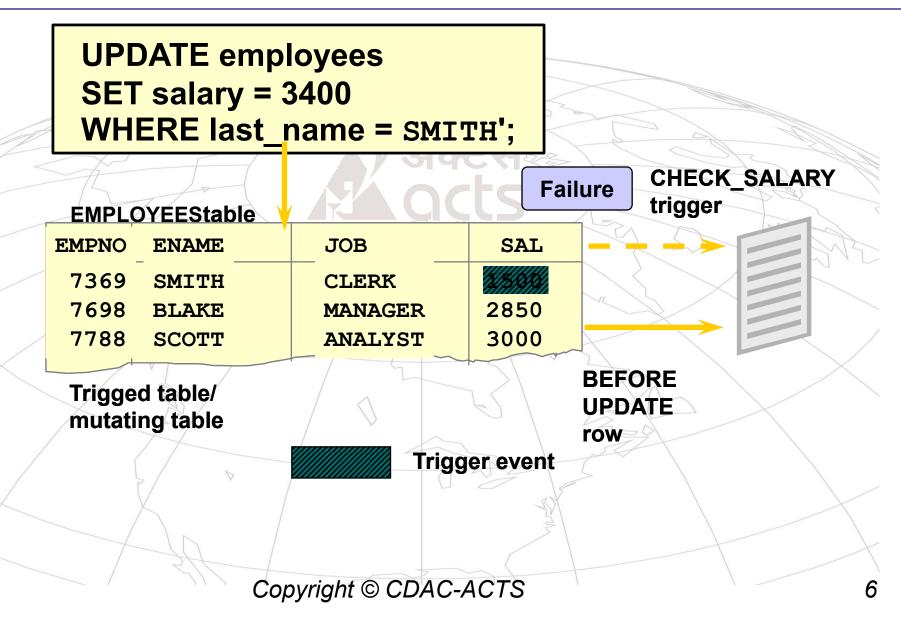
2 BEFORE INSERT ON EMP

3 CALL LOG_EXECUTION

4 /
```



Reading Data from a Mutating Table





Mutating Table: Example

```
CREATE OR REPLACE TRIGGER check salary
BEFORE INSERT OR UPDATE OF salary, job_id
ON employees
FOR EACH ROW
WHEN (NEW.job_id <> 'AD_PRES')
DECLARE
   v_minsalary employees.salary%TYPE;
   v_maxsalary employees.salary%TYPE;
BEGIN
   SELECT MIN(salary), MAX(salary)
   INTO v_minsalary, v_maxsalary
   FROM employees
   WHERE job_id = :NEW.job_id;
       :NEW.salary < v_minsalary OR
       :NEW.salary > v_maxsalary THEN
       RAISE_APPLICATION_ERROR(-20505,'Out of range');
   END IF:
END;
```

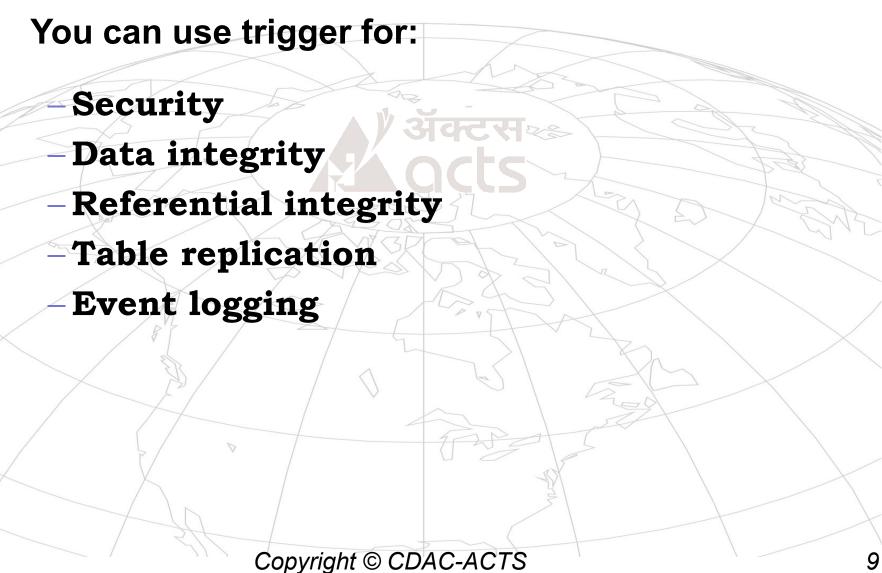


Mutating Table: Example

```
UPDATE employees
SET salary = 3400
WHERE last name = 'Stiles';
ERROR at line 2:
ORA-04091: table EMPLOYEES is mutating,
trigger/function may not see it
ORA-06512: at "CHECK SALARY", line 5
ORA-04088: error during execution of trigger
 'CHECK SALARY'
```



Implementation of Triggers





Controlling Security Within the Server



TO clerk; -- database role

GRANT clerk TO scott;



Controlling Security with a Database Trigger

```
CREATE OR REPLACE TRIGGER secure emp
BEFORE INSERT OR UPDATE OR DELETE ON employees
DECLARE
      v dummy VARCHAR2(1);
BEGIN
   IF (TO CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN'))
   THEN
      RAISE APPLICATION ERROR (-20506, 'You may only
      change data during normal business hours.');
   END IF:
   SELECT COUNT(*) INTO v_dummy FROM holiday
   WHERE holiday date = TRUNC (SYSDATE);
   IF v dummy > 0 THEN
      RAISE_APPLICATION_ERROR(-20507,
      'You may not change data on a holiday.');
   END IF;
END;
```



Auditing Using the Server Facility

AUDIT INSERT, UPDATE, DELETE ON departments BY ACCESS WHENEVER SUCCESSFUL;

Oracle will store the audit information in a data dictionary table.



Auditing Using a Trigger

```
CREATE OR REPLACE TRIGGER audit emp values
AFTER DELETE OR INSERT OR UPDATE ON employees
FOR EACH ROW
BEGIN
   IF (audit_emp_package.g_reason IS NULL) THEN
      RAISE APPLICATION ERROR (-20059, 'Specify a reason
      for the data operation through the procedure SET REASON
      of the AUDIT EMP PACKAGE before proceeding.');
   ELSE
      INSERT INTO audit_emp_table (user_name, timestamp, id,
      old_last_name, new_last_name, old_title, new_title,
      old_salary, new_salary, comments)
      VALUES (USER, SYSDATE, :OLD.employee id, :OLD.last name,
      :NEW.last_name, :OLD.job_id, :NEW.job_id, :OLD.salary,
      :NEW.salary, audit emp package.g reason);
   END IF;
END;
```



Enforce Data Integrity Within the Server







Protect Data Integrity with a Trigger

```
CREATE OR REPLACE TRIGGER check_salary
BEFORE UPDATE OF salary ON employees
FOR EACH ROW
WHEN (NEW.salary < OLD.salary)
BEGIN
RAISE_APPLICATION_ERROR (-20508,
'Do not decrease salary.');
END;
/
```



Enforce Referential Integrity Within the Server

ALTER TABLE employees
ADD CONSTRAINT emp_deptno_fk
FOREIGN KEY (department_id)
REFERENCES departments(department_id)
ON DELETE CASCADE;



Protect Referential Integrity with a Trigger

```
CREATE OR REPLACE TRIGGER cascade_updates
AFTER UPDATE OF department_id ON departments
FOR EACH ROW
BEGIN
  UPDATE employees
  SET employees.department_id=:NEW.department_id
  WHERE employees.department id=:OLD.department id;
  UPDATE job history
  SET department_id=:NEW.department_id
  WHERE department_id=:OLD.department_id;
END;
```



Trigger Information

You can view the following trigger information:

- USER_OBJECTS data dictionary view: object information
- USER_TRIGGERS data dictionary view: the text of the trigger
- USER_ERRORS data dictionary view: PL/SQL syntax errors (compilation errors) of the trigger



Benefits of Database Triggers

- Improved data security:
- Provide value-based security checks
- Provide value-based auditing
- Improved data integrity:
 - Enforce dynamic data integrity constraints
- -Enforce complex referential integrity constraints
- -Ensure related operations are performed together implicitly

Summary

- Use advanced database triggers
- List mutating and constraining rules for triggers
- Describe the real-world application of triggers
 - Auditing
 - Protecting Referential Integrity
 - Enforcing Data Integrity
 - Computing Derived Values
- Manage triggers
- View trigger information



